“Python 气象数据分析”实验报告

使用 Python 3 进行气象数据分析

实验原理

气象数据是在网上很容易找到的一类数据。很多网站都提供以往的气压、气温、湿度和降雨量等气象数据。只需指定位置和日期，就能获取一个气象数据文件。这些测量数据是由气象站收集的。气象数据这类数据源涵盖的信息范围较广。数据分析的目的是把原始数据转化为信息，再把信息转化为知识，因此拿气象数据作为数据分析的对象来讲解数据分析全过程再合适不过。

待检验的假设：靠海对气候的影响

夏天酷热难耐，住在大城市的人感受更为强烈。于是周末很多人到山村或海滨城市去游玩，放松一下身心，远离内陆城市的闷热天气。靠海会对气候产生什么问题呢？由此引起了“海洋对一个地区的气候有何影响”这个问题。

开发准备

数据源：亚得里亚海和波河流域的10个城市 的天气数据

选作样本的城市列表如下：

Ferrara（费拉拉） Torino（都灵） Mantova（曼托瓦） Milano（米兰） Ravenna（拉文纳） Asti（阿斯蒂） Bologna（博洛尼亚） Piacenza（皮亚琴察） Cesena（切塞纳） Faenza（法恩莎） 网址为：气象数据

```
# 下载气象数据
!wget -nc http://labfile.oss.aliyuncs.com/courses/780/WeatherData.zip

# 安装 unzip 解压缩
!apt-get install unzip

# 解压缩
!unzip -o WeatherData.zip
```

查看城市的天气数据文件：

```
!apt-get install tree

!tree WeatherData/
```

导入所需的包

```
import numpy as np
import pandas as pd
import datetime
```

读取数据

```
df_ferrara = pd.read_csv('WeatherData/ferrara_270615.csv')
df_milano = pd.read_csv('WeatherData/milano_270615.csv')
df_mantova = pd.read_csv('WeatherData/mantova_270615.csv')
df_ravenna = pd.read_csv('WeatherData/ravenna_270615.csv')
df_torino = pd.read_csv('WeatherData/torino_270615.csv')
df_asti = pd.read_csv('WeatherData/asti_270615.csv')
df_bologna = pd.read_csv('WeatherData/bologna_270615.csv')
df_piacenza = pd.read_csv('WeatherData/piacenza_270615.csv')
df_cesena = pd.read_csv('WeatherData/cesena_270615.csv')
df_faenza = pd.read_csv('WeatherData/faenza_270615.csv')
```

实验步骤

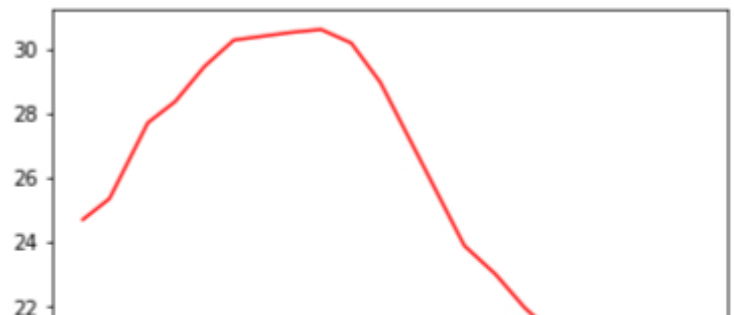
导入必要的库:

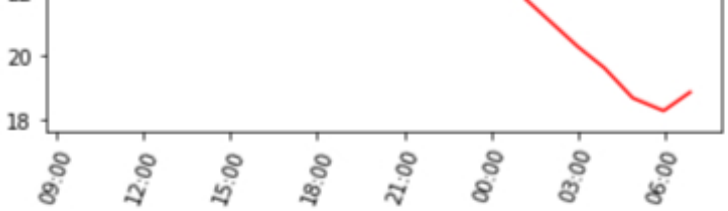
```
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from dateutil import parser
```

温度数据分析

```
# 取出我们要分析的温度和日期数据
y1 = df_milano['temp']
x1 = df_milano['day']

# 把日期数据转换成 datetime 的格式
day_milano = [parser.parse(x) for x in x1]

# 调用 subplot 函数，fig 是图像对象，ax 是坐标轴对象
fig, ax = plt.subplots()

# 调整x轴坐标刻度，使其旋转70度，方便查看
plt.xticks(rotation=70)

# 设定时间的格式
hours = mdates.DateFormatter('%H:%M')

# 设定X轴显示的格式
ax.xaxis.set_major_formatter(hours)

# 画出图像，day_milano是X轴数据，y1是Y轴数据，‘r’代表的是'red' 红色
ax.plot(day_milano ,y1, 'r')
```

```
[<matplotlib.lines.Line2D at 0x7f949dbcb668>]
```



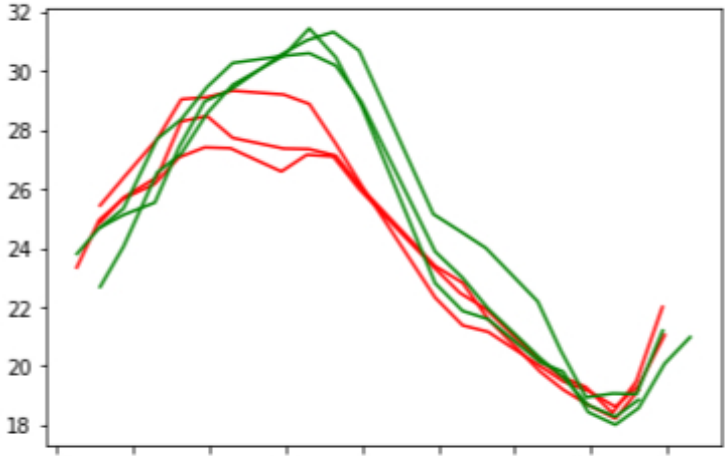


由图可见，气温走势接近正弦曲线，从早上开始气温逐渐升高，最高温出现在下午两点到六点之间，随后气温逐渐下降，在第二天早上六点时达到最低值。

分析几个不同城市的气温趋势

```
# 读取温度和日期数据
y1 = df_ravenna['temp']
x1 = df_ravenna['day']
y2 = df_faenza['temp']
x2 = df_faenza['day']
y3 = df_cesena['temp']
x3 = df_cesena['day']
y4 = df_milano['temp']
x4 = df_milano['day']
y5 = df_asti['temp']
x5 = df_asti['day']
y6 = df_torino['temp']
x6 = df_torino['day']

# 把日期从 string 类型转化为标准的 datetime 类型
day_ravenna = [parser.parse(x) for x in x1]
day_faenza = [parser.parse(x) for x in x2]
day_cesena = [parser.parse(x) for x in x3]
day_milano = [parser.parse(x) for x in x4]
day_asti = [parser.parse(x) for x in x5]
day_torino = [parser.parse(x) for x in x6]

# 调用 subplots() 函数，重新定义 fig, ax 变量
fig, ax = plt.subplots()
plt.xticks(rotation=70)

hours = mdates.DateFormatter('%H:%M')
ax.xaxis.set_major_formatter(hours)

#这里需要画出三根线，所以需要三组参数， 'g'代表'green'
ax.plot(day_ravenna,y1,'r',day_faenza,y2,'r',day_cesena,y3,'r')
ax.plot(day_milano,y4,'g',day_asti,y5,'g',day_torino,y6,'g')
```

```
[<matplotlib.lines.Line2D at 0x7f949d854518>,
 <matplotlib.lines.Line2D at 0x7f949d805f28>,
 <matplotlib.lines.Line2D at 0x7f949d811780>]
```



离海最近的三个城市的气温曲线使用红色，而离海最远的三个城市的曲线使用绿色。 如图 所示，结果看起来不错。离海最近的三个城市的最高气温比离海最远的三个城市低不少，而最低气温看起来差别较小。

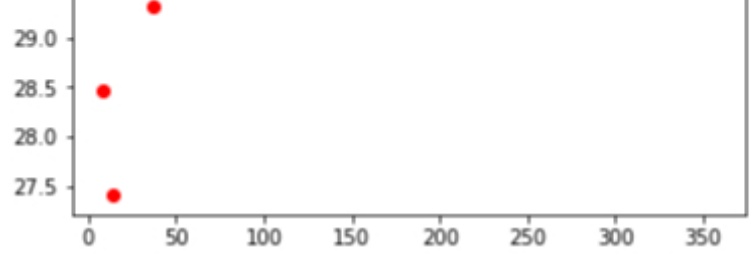
用线性图表示温度最值点和离海远近之间的关系

```
# dist 是一个装城市距离海边距离的列表
dist = [df_ravenna['dist'][0],
    df_cesena['dist'][0],
    df_faenza['dist'][0],
    df_ferrara['dist'][0],
    df_bologna['dist'][0],
    df_mantova['dist'][0],
    df_piacenza['dist'][0],
    df_milano['dist'][0],
    df_asti['dist'][0],
    df_torino['dist'][0]
]

# temp_max 是一个存放每个城市最高温度的列表
temp_max = [df_ravenna['temp'].max(),
    df_cesena['temp'].max(),
    df_faenza['temp'].max(),
    df_ferrara['temp'].max(),
    df_bologna['temp'].max(),
    df_mantova['temp'].max(),
    df_piacenza['temp'].max(),
    df_milano['temp'].max(),
    df_asti['temp'].max(),
    df_torino['temp'].max()
]

# temp_min 是一个存放每个城市最低温度的列表
temp_min = [df_ravenna['temp'].min(),
    df_cesena['temp'].min(),
    df_faenza['temp'].min(),
    df_ferrara['temp'].min(),
    df_bologna['temp'].min(),
    df_mantova['temp'].min(),
    df_piacenza['temp'].min(),
    df_milano['temp'].min(),
    df_asti['temp'].min(),
    df_torino['temp'].min()
]
```

画出最高温

```
fig, ax = plt.subplots()
ax.plot(dist,temp_max,'ro')
```

```
[<matplotlib.lines.Line2D at 0x7f9488f4cef0>]
```



由此可以推知，海洋对气象数据具有一定程度的影响这个假设是正确的（至少这一天如此）

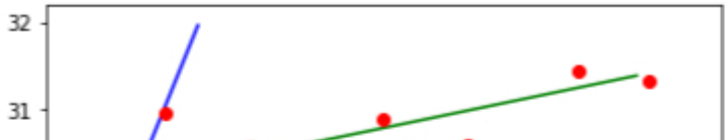
用回归算法得到两条直线，分别代表两种不同的气温趋势

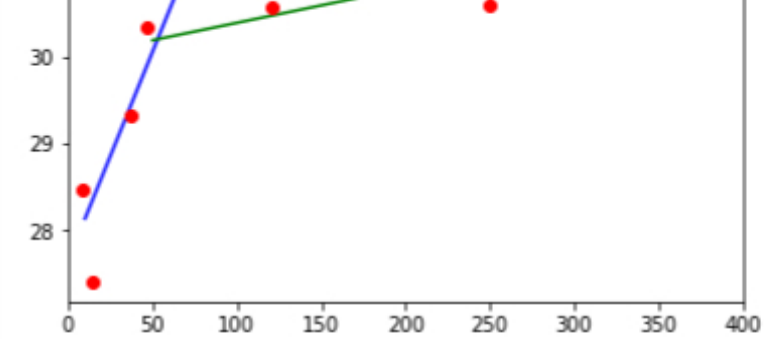
```
from sklearn.svm import SVR

# dist1是靠近海的城市集合，dist2是远离海洋的城市集合
dist1 = dist[0:5]
dist2 = dist[5:10]

# 改变列表的结构，dist1现在是5个列表的集合
# 之后我们会看到 nbumpy 中 reshape() 函数也有同样的作用
dist1 = [[x] for x in dist1]
dist2 = [[x] for x in dist2]

# temp_max1 是 dist1 中城市的对应最高温度
temp_max1 = temp_max[0:5]
# temp_max2 是 dist2 中城市的对应最高温度
temp_max2 = temp_max[5:10]

# 我们调用SVR函数，在参数中规定了使用线性的拟合函数
# 并且把 C 设为1000来尽量拟合数据（因为不需要精确预测不用担心过拟合）
svr_lin1 = SVR(kernel='linear', C=1e3)
svr_lin2 = SVR(kernel='linear', C=1e3)

# 加入数据，进行拟合（这一步可能会跑很久，大概10多分钟，休息一下:））
svr_lin1.fit(dist1, temp_max1)
svr_lin2.fit(dist2, temp_max2)

# 关于 reshape 函数请看代码后面的详细讨论
xp1 = np.arange(10,100,10).reshape((9,1))
xp2 = np.arange(50,400,50).reshape((7,1))
yp1 = svr_lin1.predict(xp1)
yp2 = svr_lin2.predict(xp2)
```

绘图

```
# 限制了 x 轴的取值范围
fig, ax = plt.subplots()
ax.set_xlim(0,400)

# 画出图像
ax.plot(xp1, yp1, c='b', label='Strong sea effect')
ax.plot(xp2, yp2, c='g', label='Light sea effect')
ax.plot(dist,temp_max,'ro')
```

```
[<matplotlib.lines.Line2D at 0x7f9488fda128>]
```





如上所见，离海 60 公里以内，气温上升速度很快，从 28 度陡升至 31 度，随后增速渐趋缓和（如果还继续增长的话），更长的距离才会有小幅上升。这两种趋势可分别用两条直线来表示，直线的表达式为： y=ax+b

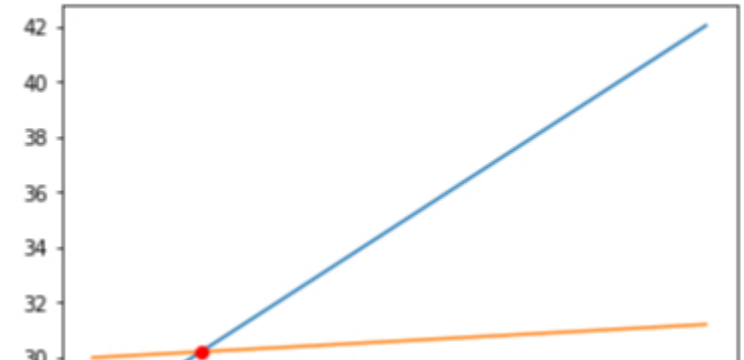
```
print(svr_lin1.coef_)  #斜率
print(svr_lin1.intercept_)  # 截距
print(svr_lin2.coef_)
print(svr_lin2.intercept_)
```

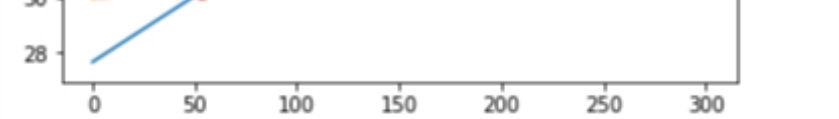
求两条直线的交点

```
from scipy.optimize import fsolve

# 定义了第一条拟合直线
def line1(x):
    a1 = svr_lin1.coef_[0][0]
    b1 = svr_lin1.intercept_[0]
    return a1*x + b1

# 定义了第二条拟合直线
def line2(x):
    a2 = svr_lin2.coef_[0][0]
    b2 = svr_lin2.intercept_[0]
    return a2*x + b2

# 定义了找到两条直线的交点的 x 坐标的函数
def findIntersection(fun1,fun2,x0):
    return fsolve(lambda x : fun1(x) - fun2(x),x0)

result = findIntersection(line1,line2,0.0)
print("[x,y] = [ %d , %d ]" % (result,line1(result)))

# x = [0,10,20, ..., 300]
x = np.linspace(0,300,31)
plt.plot(x,line1(x),x,line2(x),result,line1(result),'ro')
```

```
[x,y] = [ 53 , 30 ]
```

```
[<matplotlib.lines.Line2D at 0x7f9488e49c88>,
 <matplotlib.lines.Line2D at 0x7f9488e49e48>,
 <matplotlib.lines.Line2D at 0x7f9488e526a0>]
```



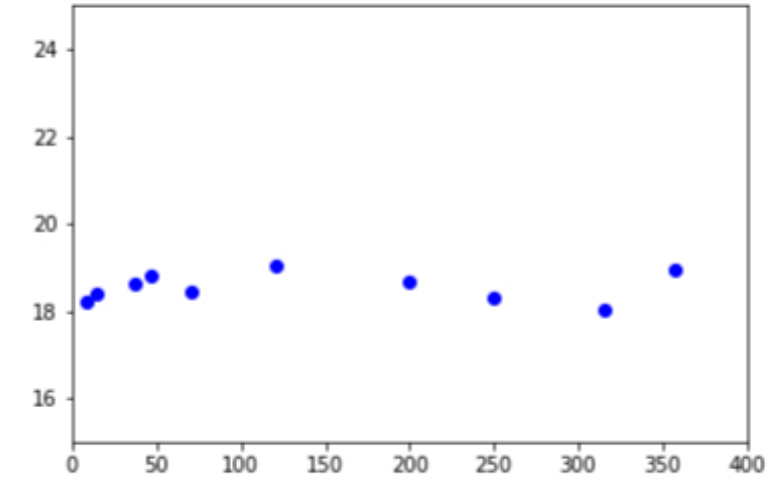


因此，可以说海洋对气温产生影响的平均距离（该天的情况）为 53 公里。

分析最低温度

```
# axis 函数规定了 x 轴和 y 轴的取值范围
plt.axis((0,400,15,25))
plt.plot(dist,temp_min,'bo')
```

```
[<matplotlib.lines.Line2D at 0x7f9488d047b8>]
```

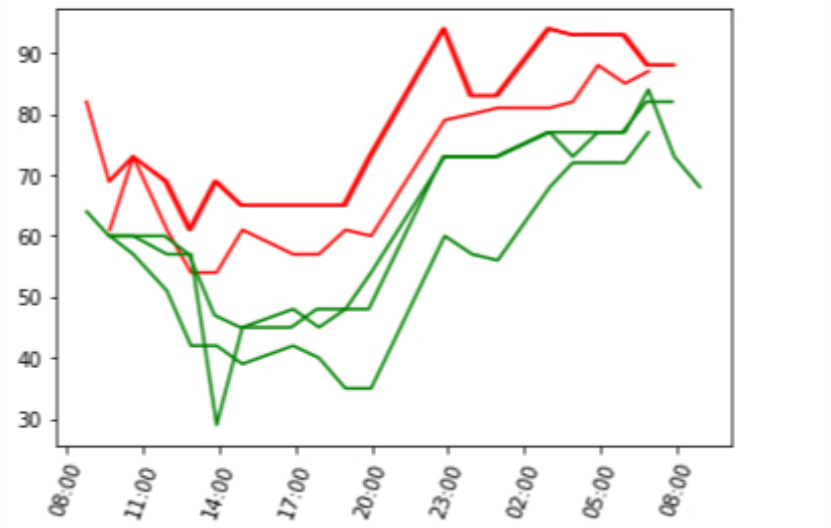


在这个例子中，很明显夜间或早上6点左右的最低温与海洋无关。

湿度数据分析

```
# 读取湿度数据
y1 = df_ravenna['humidity']
x1 = df_ravenna['day']
y2 = df_faenza['humidity']
x2 = df_faenza['day']
y3 = df_cesena['humidity']
x3 = df_cesena['day']
y4 = df_milano['humidity']
x4 = df_milano['day']
y5 = df_asti['humidity']
x5 = df_asti['day']
y6 = df_torino['humidity']
x6 = df_torino['day']

# 重新定义 fig 和 ax 变量
fig, ax = plt.subplots()
plt.xticks(rotation=70)

# 把时间从 string 类型转化为标准的 datetime 类型
day_ravenna = [parser.parse(x) for x in x1]
day_faenza = [parser.parse(x) for x in x2]
day_cesena = [parser.parse(x) for x in x3]
day_milano = [parser.parse(x) for x in x4]
day_asti = [parser.parse(x) for x in x5]
day_torino = [parser.parse(x) for x in x6]

# 规定时间的表示方式
hours = mdates.DateFormatter('%H:%M')
```

```
ax.xaxis.set_major_formatter(hours)

#表示在图上
ax.plot(day_ravenna,y1,'r',day_faenza,y2,'r',day_cesena,y3,'r')
ax.plot(day_milano,y4,'g',day_asti,y5,'g',day_torino,y6,'g')
```

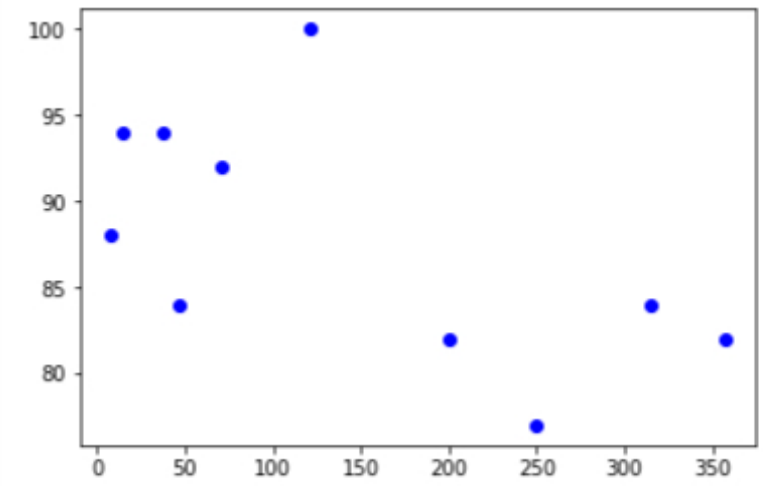


看上去好像近海城市的湿度要大于内陆城市，全天湿度差距在 20% 左右。

分析湿度的极值和离海远近的关系

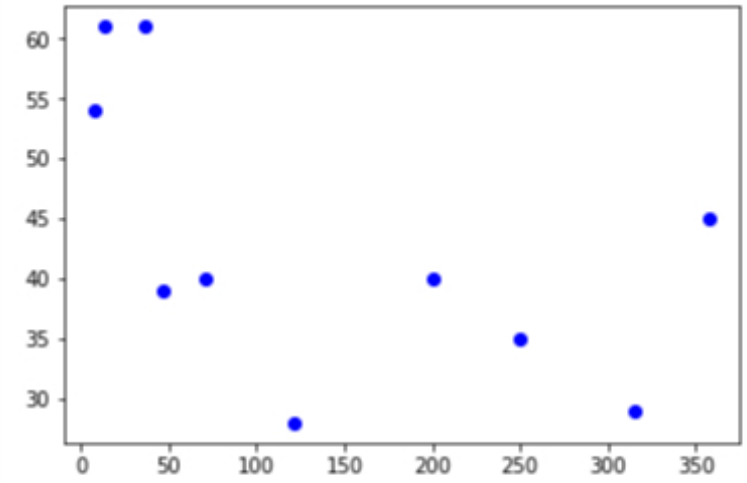
```
# 获取最大湿度数据
hum_max = [df_ravenna['humidity'].max(),
df_cesena['humidity'].max(),
df_faenza['humidity'].max(),
df_ferrara['humidity'].max(),
df_bologna['humidity'].max(),
df_mantova['humidity'].max(),
df_piacenza['humidity'].max(),
df_milano['humidity'].max(),
df_asti['humidity'].max(),
df_torino['humidity'].max()
]

plt.plot(dist,hum_max,'bo')
```

```
[<matplotlib.lines.Line2D at 0x7f9488c32e10>]
```




```
# 获取最小湿度
hum_min = [
```

```
df_ravenna['humidity'].min(),
df_cesena['humidity'].min(),
df_faenza['humidity'].min(),
df_ferrara['humidity'].min(),
df_bologna['humidity'].min(),
df_mantova['humidity'].min(),
df_piacenza['humidity'].min(),
df_milano['humidity'].min(),
df_asti['humidity'].min(),
df_torino['humidity'].min()
]
plt.plot(dist,hum_min,'bo')
```



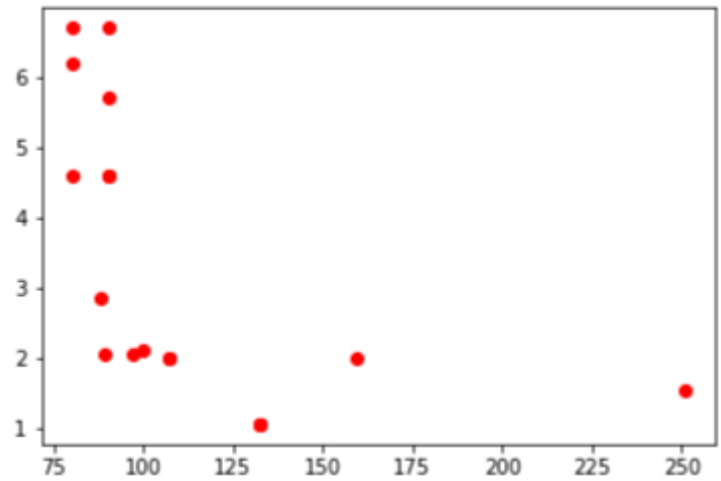
由图 可以确定，近海城市无论是最大还是最小湿度都要高于内陆城市。然而,在我看来，我们还不能说湿度和距离之间存在线性关系或者其他能用曲线表示的关系。原因是采集到的数据点数量太少，不足以描述这类趋势。

风向频率玫瑰图

在我们采集的每个城市的气象数据中，下面两个与风有关： 风力（风向） 风速 分析存放每个城市气象数据的 DataFrame 就会发现，风速不仅跟一天的时间段相关联，还与一个介于 0~360 度的方向有关。

把一个DataFrame中的数据点做成散点图

```
plt.plot(df_ravenna['wind_deg'],df_ravenna['wind_speed'],'ro')
```

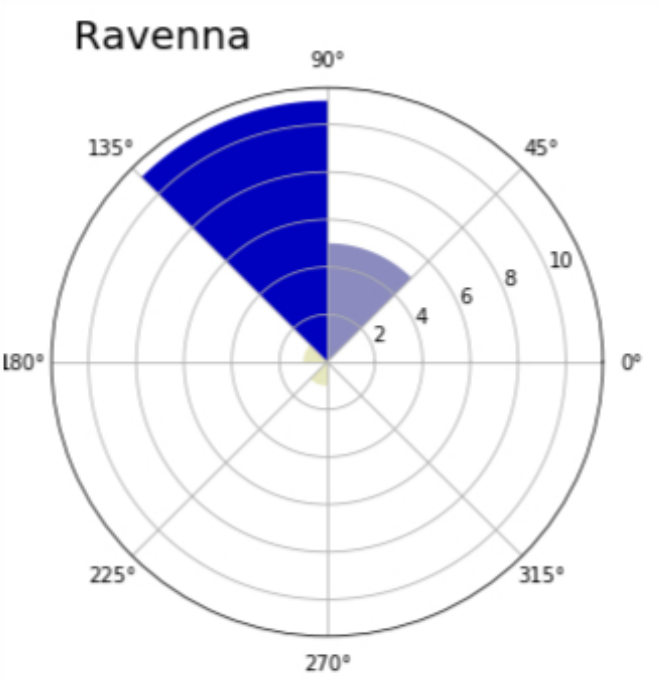


但显然该图的表现力不足 所以要创建另一种可视化方法：极区图

```
hist, bins = np.histogram(df_ravenna['wind_deg'],8,[0,360])
print(hist)
```

```
print(bins)

def showRoseWind(values,city_name,max_value):
    N = 8

    # theta = [pi*1/4, pi*2/4, pi*3/4, ..., pi*2]
    theta = np.arange(2 * np.pi / 16, 2 * np.pi, 2 * np.pi / 8)
    radii = np.array(values)
    # 绘制极区图的坐标系
    plt.axes([0.025, 0.025, 0.95, 0.95], polar=True)

    # 列表中包含的是每一个扇区的 rgb 值，x越大，对应的color越接近蓝色
    colors = [(1-x/max_value, 1-x/max_value, 0.75) for x in radii]

    # 画出每个扇区
    plt.bar(theta, radii, width=(2*np.pi/N), bottom=0.0, color=colors)

    # 设置极区图的标题
    plt.title(city_name, x=0.2, fontsize=20)
```

```
showRoseWind(hist,'Ravenna',max(hist))
```

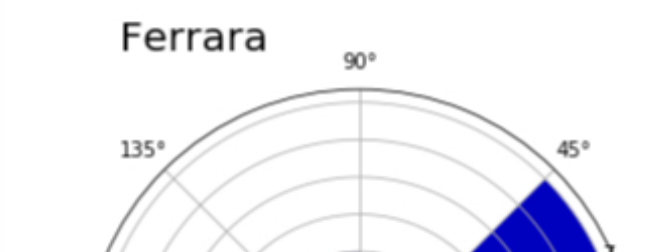


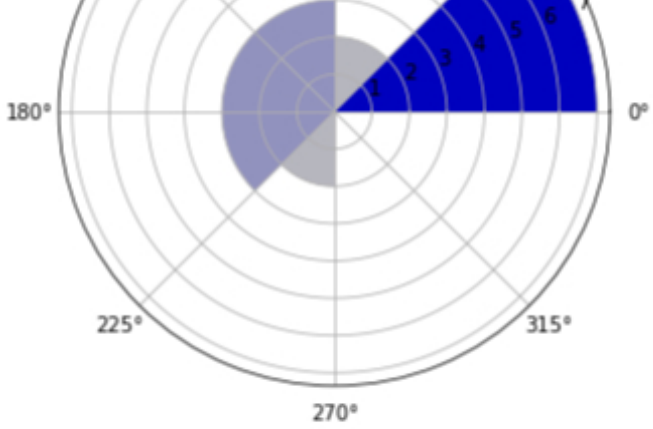
由图 可见，整个 360 度的范围被分成八个区域（面元），每个区域弧长为 45 度，此外每个区域还有一列呈放射状排列的刻度值。在每个区域中，用半径长度可以改变的扇形表示一个数值，半径越长，扇形所表示的数值就越大。为了增强图表的可读性，我们使用与扇形半径相对应的颜色表。半径越长，扇形跨度越大，颜色越接近于深蓝色。

从刚得到的极区图可以得知风向在极坐标系中的分布方式。该图表示这一天大部分时间风都吹向西南和正西方向。

查看其他城市的风向情况

```
hist, bin = np.histogram(df_ferrara['wind_deg'],8,[0,360])
print(hist)
showRoseWind(hist,'Ferrara',max(hist))
```



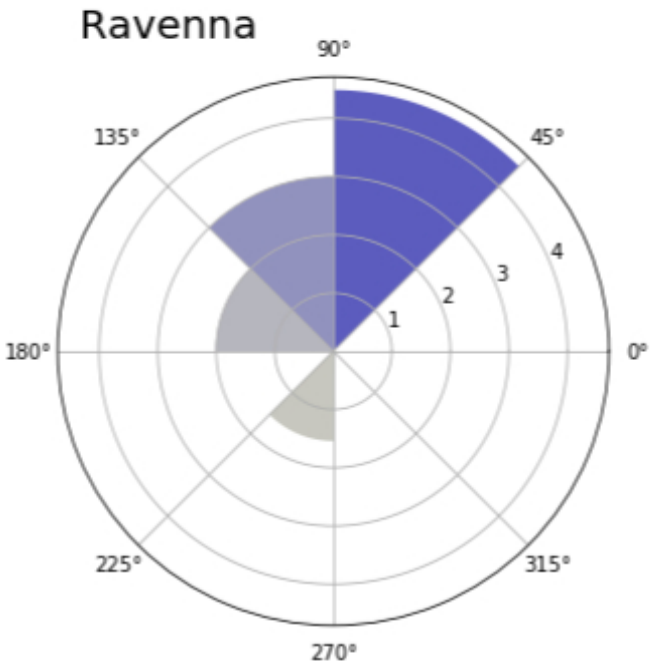


计算风速均值的分布情况

```
def RoseWind_Speed(df_city):
    # degs = [45, 90, ..., 360]
    degs = np.arange(45,361,45)
    tmp = []
    for deg in degs:
        # 获取 wind_deg 在指定范围的风速平均值数据
        tmp.append(df_city[(df_city['wind_deg']>(deg-46)) & (df_city['wind_deg']<deg)]
        ['wind_speed'].mean())
    return np.array(tmp)
```

调用函数showRoseWind()

```
showRoseWind(RoseWind_Speed(df_ravenna),'Ravenna',max(hist))
```



实验总结

本次的实验项目用到了Python中的numpy库、matplotlib库、pandas库，并用sklearn中的SVR（）函数，用回归算法进行对比分析温度的变化，使本人加深了对于这些库的理解以及应用。由于数据量不多，并不能具有很强的代表性，只是熟悉了数据分析的基本流程。